International Journal of Information and Computer Science
IJICS Volume 1, Issue 7, October 2012 PP. 163-177
ISSN(online) 2161-5381
ISSN(print) 2161-6450
www.iji-cs.org

Hexagon-based Range-free Localization for Large-scale Environmental Sensor Networks

Eva M. García[1], M. Ángeles Serna[1], Antonio Robles-Gómez[2], Aurelio Bermúdez[1], Rafael Casado[1]
[1]Instituto de Investigación en Informática de Albacete (I3A), Universidad de Castilla-La Mancha, Albacete, Spain
[2]Escuela Técnica Superior de Ingeniería Informática, Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain
Email: aurelio.bermudez@uclm.es

Abstract. Most of the applications of sensor networks require sensors being aware of their position. Usually, this position is estimated by means of a distributed localization algorithm, which assumes that the position of some network nodes is known a priori. In many cases each node obtains the area where it resides by intersecting the coverage areas of the nodes that it hears. Using circles to model these coverage areas introduces a high computational complexity. For this reason, some authors have modeled these areas by means of squares. In this paper we propose the employment of hexagons, in order to reduce the additional inaccuracy introduced by that shape.

Keywords: Wireless Sensor Network; Distributed Localization; Range-Free; Performance Evaluation

1. INTRODUCTION

Nowadays, wireless sensor networks (WSNs) are an active research field because of their wide range of applications, which include disaster relief operation, biodiversity mapping, and intelligent buildings [1]. The performance of a deployed WSN is widely influenced by sensor localization information, that is, each sensor composing the network needs to know its geographical position. Constraints of size, energy consumption, and price make it unfeasible to equip every node in the network with a GPS (*Global Positioning System*) receiver [2]. However, it may be reasonable to incorporate a GPS into a small subset of the sensors. Such nodes, referred to as beacons in the literature, may be used to help estimate the position of the rest of the nodes.

There are two general groups of localization techniques: *range-free* and *range-based*. A range-based technique estimates the position of a node starting from its distance to several beacon nodes. *Time difference of arrival* (TDOA) [3], [4], [5], *angle of arrival* (AOA) [6], [7], [8], and *received signal strength* (RSS) [9] are the most popular methods to measure the distance between nodes. These techniques have two main drawbacks. First, sensor nodes require expensive additional hardware. Secondly, the accuracy of the measurement can be affected by environmental interferences. These issues make range-based techniques unsuitable for localization in a large-scale environmental sensor network.

On the other hand, in a range-free technique it is not necessary to estimate distances to beacon nodes. Instead, each sensor node uses the information received from a few neighbor nodes (beacons or not) to calculate its own approximate location. These techniques assume that a sensor node is located inside the overlapping coverage area of its direct neighbors. The way to model the coverage and the estimated localization area has brought about in recent years the development of several localization algorithms. Algorithms based on circular intersection are located at one end, providing great accuracy, but requiring the use of trigonometric functions and complex data structures. At the other end, we have algorithms based on rectangular intersection, with very low computational cost and more imprecision in estimates.

In this work we describe in detail a solution that establishes a trade-off between both points: hexagonal intersection. The use of hexagons provides more accuracy than squares on devices localization. In previous works [10], it has been demonstrated that the hexagonal intersection achieves an improvement of 12 percent in comparison with rectangular intersection. Another advantage of the hexagonal intersection proposal is that the shape obtained when two hexagons are intersected can be defined by means of only three points, while the result of intersecting two circular areas is an irregular shape, which requires a complex data structure to be stored or transmitted. Moreover, if the intersection process is performed in an iterative fashion, the resulting areas always require three points in the case of hexagonal intersection, while the data structure required by circular intersection is more and more complex.

The evaluation of the hexagonal intersection technique performed in [10] was done by using networks composed up to of 300 sensor nodes. Furthermore, a nonrealistic simulator was employed, in which important issues like signal attenuation and radio frequency noise were not considered. In this work we evaluate the behavior of our proposal by using large-scale networks and a more realistic simulation environment.

The paper is organized as follows. The next section presents some related work in the area of sensor node localization by means of range-free techniques. Section 3 describes the proposed algorithm and details the basics of hexagonal intersection. Section 4 describes the simulation tool developed in order to evaluate the behavior of localization algorithms. A comparative performance analysis between rectangular and hexagonal intersection is provided in Section 5. Lastly, in Section 6 some final conclusions and possible future lines of work are offered.

2. RELATED WORK

A pioneer work in the category of range-free localization techniques is the *Centroid* algorithm [11]. In this proposal, initially each node collects information from the set of beacons that it hears. Then, starting from the location of the beacons, the node estimates its own location by means of the expression:

$$(X_{est}, Y_{est}) = \left(\frac{X_1 + ...X_N}{N}, \frac{Y_1 + ...Y_N}{N} \right)$$

where N is the number of beacons and (X_i, Y_i) their respective locations.

However, although this is a very simple approach, in some cases it obtains an estimation located outside the beacons overlapping area.

Several range-free techniques are based on *rectangular intersection*. Basically, these proposals assume that if a node *A* can hear the transmission of a node *B*, *A* is located in some point inside a square that is centered at *B*. The side length of this square is twice the radio range of the second node. An example is the *Bounding-Box* algorithm [12], in which each node collects the position of its neighboring beacons and then obtains the intersection of the squares centered at these locations. Obviously, the result of this simple operation is a rectangle, whose center is the final estimation that the algorithms produces (see Figure 1).

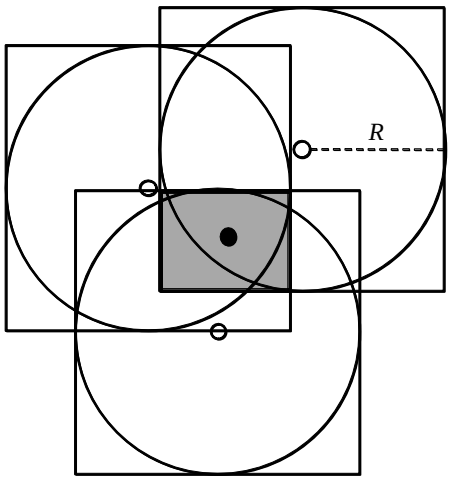


Figure 1. Rectangular Intersection. Beacons know their exact position. The central node estimates its position from intersection of squares representing beacons range.

Two distributed and iterative versions of the rectangular intersection technique have been proposed in [13] and [14]. These works consider the existence of nodes not covered by the beacons. In this case, the localization process is started by the beacons, which transmit their position to the medium. Then, the receiving nodes extend these positions by using the radio range, and obtain the corresponding overlapping rectangle. After that, the new estimation is disseminated again, in an iterative way. The process ends when either the estimation reaches a tolerance value previously established or the time for the localization task expires. Two mechanisms for reducing the huge control overhead inherent to this process have been recently proposed in [15] and [16].

The DRLS (*Distributed Range-Free Localization Scheme*) technique [17], more recent that the previous ones, is based on the same principles. This proposal adds a refinement step after the rectangular intersection process, in which the resulting rectangle is seen as a grid. Then, the number of beacons that are heard in each cell is considered. The algorithm assumes that the cells with a higher value define the area in which the probability of finding the node is higher.

The DLE (*Distributed Location Estimating*) algorithm [18] also performs a rectangular intersection. In this technique, for each pair of beacons in a given localization, an area known as *Estimative Rectangle* (ER) is computed, starting from the intersection points of the circles defined by the beacons. The final estimation is obtained by intersecting all the ERs previously computed.

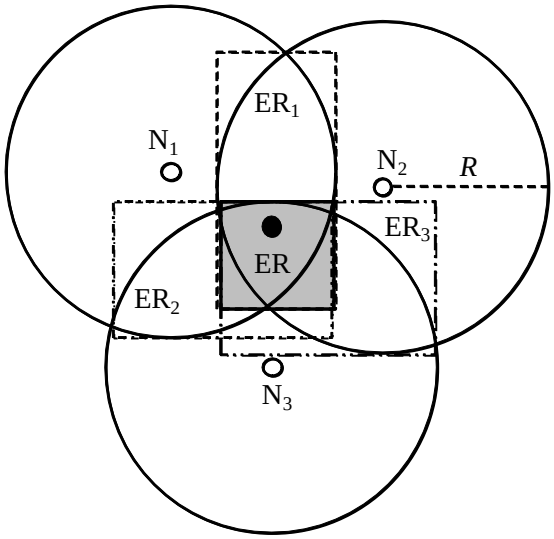


Figure 2. Obtaining an estimation from three overlapping ERs in DLE.

Another popular range-free approximation for locating sensor nodes is the APIT (*Approximate Point in Triangulation*) technique [19]. It defines a set of triangular areas in the localization area, starting from the location of the beacon nodes (see Figure 3). The presence of a node inside or outside these triangular regions allows an area to be defined, in which the node could reside. The method used for defining this area

is called *Test PIT*. In this test, a node chooses three beacons from the set of beacons that it can hear. Then, it tests if it is located inside the triangle defined by these beacons. APIT repeats this test by using different beacon combinations. It then computes the center of gravity for the intersection of all the triangles in which the node resides. Obviously, the APIT method requires a large number of beacon references, which can be provided, for example, for a mobile beacon.

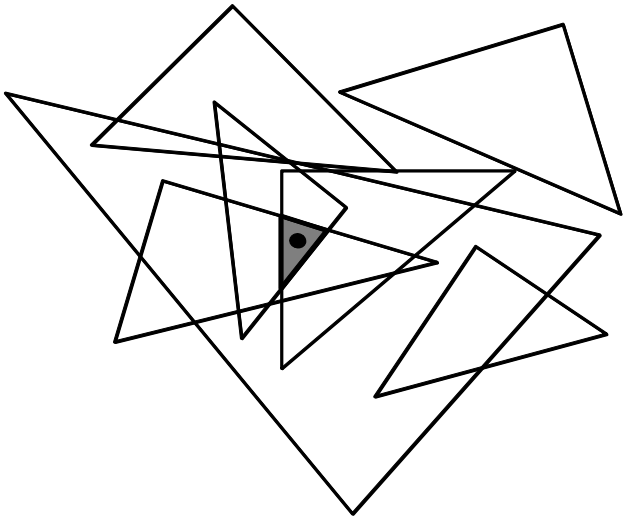


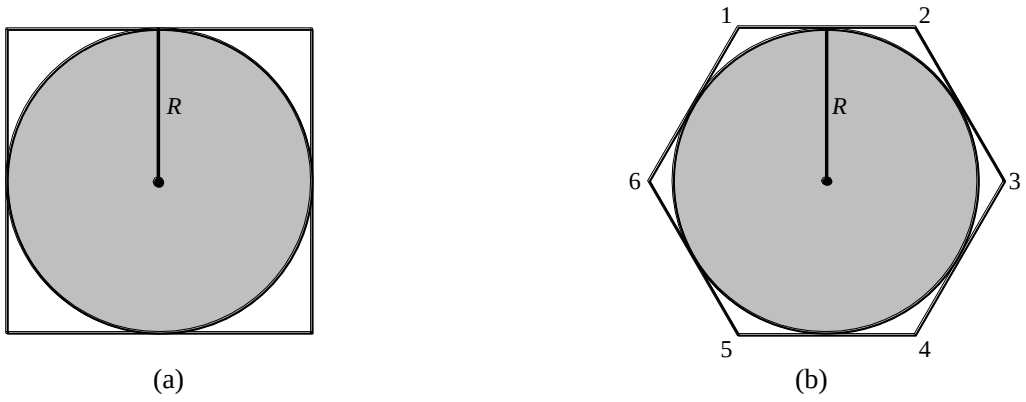
Figure 3. APIT method.

Finally, several works propose the use of mobile nodes which collaborate in the localization task. For example, in [14] –a technique based on rectangular intersection– authors propose that sensors can use the observations made of a mobile target in order to improve the estimations for the localization of both the mobile target and the network sensors. The algorithm proposed in [20] employs mobile beacons moving over the deployment area and transmitting their current locations. A sensor that receives these beacon messages estimates its own position by using a basic geometric rule: a perpendicular bisector of a chord passes through the center of the circle. Assuming that the sensor transmission range defines a circle and that we can know two chords in it from the beacon messages, then the intersection of the corresponding perpendicular bisectors gives the node position (that is, the center of the circle). This strategy has the disadvantage that it requires the mobile beacon to go through the node coverage area at least twice. The technique presented in [21] also proposes the employ of mobile beacons transmitting their position periodically. However, in this case it is only necessary for the beacons to go through the deployment area following a straight line. Once a node has received several consecutive messages from the beacon, it can determine a set of positions in its path, named as *prearrival position*, *arrival position*, *departure position*, and *postdeparture position*. Starting from the circular areas centered in these points, the node can compute a reduced region in which it is located.

3. LOCALIZATION MECHANISM BASED ON HEXAGONAL INTERSECTION

This section presents a range-free distributed localization algorithm called *hexagonal intersection*. As we have seen in the previous section, localization techniques based on rectangular intersection offer the advantage of requiring reduced computational capacity of the network nodes. However, it is obvious that the error introduced in estimates is greater than using circular coverage areas. Given that the area of the circle is πR^2 and the area of the square is $(2R)^2$, then the initial assumption of approximating a circular area by means of the square containing it (Figure 4a), involves increasing the working area by 27.3 percent.

An intermediate approach regarding the obtained error and the complexity of the computations may consist in representing coverage area by using hexagons. In particular, we start from the regular hexagon centered at the circular range whose apothem is equal to the radio range (Figure 4b).



(a) (b)

Figure 4. Approximation of a circular coverage area by using: (a) a square; (b) a hexagon.

Bearing in mind that the areas of the circle and the hexagon fit the following formulae:

$$A_{circle} = \pi r^2$$

$$A_{hexagon} = \frac{6r^2}{\sqrt{3}}$$

then, the use of this geometric shape only implies an increase of 10.2 percent in the localization area.

Although the result of intersecting rectangular areas in an iterative fashion is always a new rectangle, intersecting hexagonal areas will result in irregular polygons with 6, 5, 4, or even 3 sides (Figure 5). However, these polygons have the property that each side has a slope of 0°, 60° or 120°. Although they have not necessarily six sides, from now on we will call these polygons *pseudo-hexagons*.

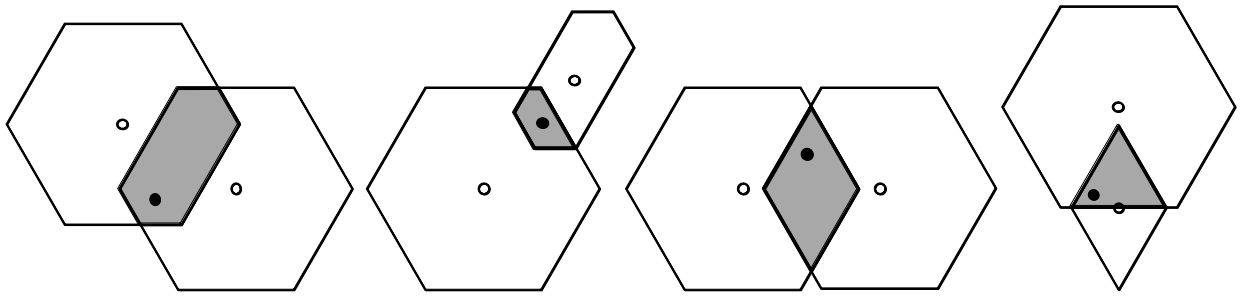


Figure 5. Some examples of hexagonal intersection.

A pseudo-hexagon can be defined by means of three vertices belonging to different sides. If we number vertices in a clockwise direction, starting with the upper left vertex (Figure 4b), we have considered vertices 1, 3, and 5.

Next, we present the proposed localization algorithm and then detail the way in which the hexagonal intersection is performed.

3.1. Localization Algorithm

In the start phase of the localization process, each node must initialize the pseudo-hexagon determining its localization. As previously stated, the coverage area of a node is determined by the regular hexagon centered at the circular range whose apothem is equal to the radio range. A node with GPS knows its exact position and, therefore, the three vertices defining its pseudo-hexagon match. In contrast, a node without localization information starts with an infinite pseudo-hexagon. After defining a starting pseudo-hexagon, each node initializes a local timer, which determines the time interval it uses to transmit localization packets.

In each step of the process, nodes having localization information (beacons or not) are responsible for sending out their estimations by means of packets. When a node receives a localization packet, it assumes that it is placed somewhere within the coverage area of the node that sent the packet. It then refines its own estimation by carrying out the intersection between its previous pseudo-hexagon and the one received, but extending it by a factor equal to its radio range. The process iterates either until a predefined threshold is reached, or for a pre-established period of time. Figure 6 shows the localization process for some node *N* (beacon or not).

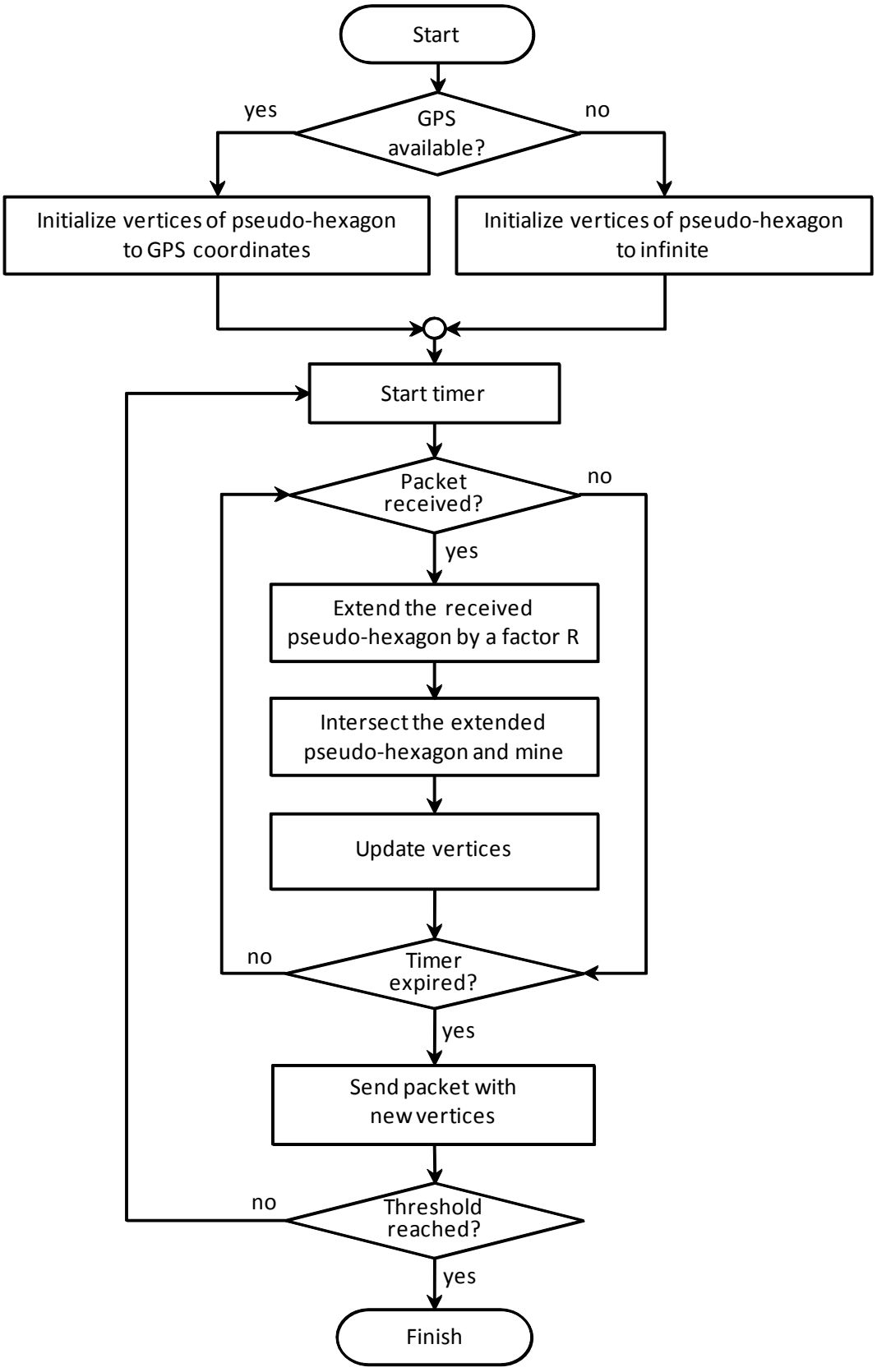


Figure 6. Localization algorithm based on hexagonal intersection.

3.2. Pseudo-hexagon Intersection

In this section we provide the mathematical basis and detail the process of hexagonal intersection.

Conceptually, the intersection of rectangles and the intersection of pseudo-hexagons are very similar. The intersection of two rectangles *A* and *B* consists in comparing each side of *A* with the corresponding side of *B*, and selecting those close to the center. After that, the computation of the two cut points of the lines defining the selected sides is performed. The fact that the sides are parallel to the coordinate axes means that the computation of cut points is implicit.

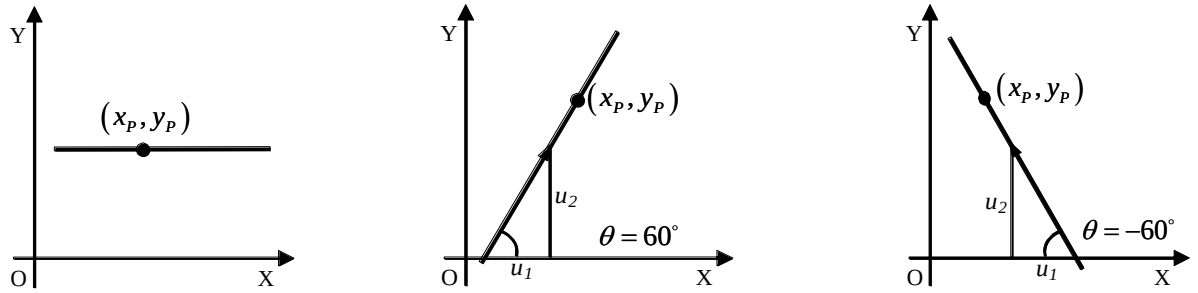
The intersection of pseudo-hexagons is carried out in the same way; however, in this case, it is necessary to compare six pairs of sides (instead of four) and also to obtain nine cut points (instead of two).

In an analogous way to rectangular intersection, in order to obtain the intersection of two pseudo-hexagons, we just have to calculate vertices 1, 3, and 5 of the resulting area. Before explaining in detail the process of intersecting two pseudo-hexagons, we expound some previous geometric basics.

3.2.1. Intersection of lines with slopes of 0°, 60°, and 120°

Given that the equations of lines with slopes of 0°, 60°, and 120° (Figure 7) are respectively:

- $y = y_P$
- $y = \sqrt{3} \cdot (x - x_P) + y_P$
- $y = \sqrt{3} \cdot (x_P - x) + y_P$



(a) 0° (b) 60° (c) 120°

Figure 7. Lines with different slopes.

Then, we can solve the equations system defining the intersection (x_I, y_I) of lines with slopes of 0° and 60° (see Figure 8a):

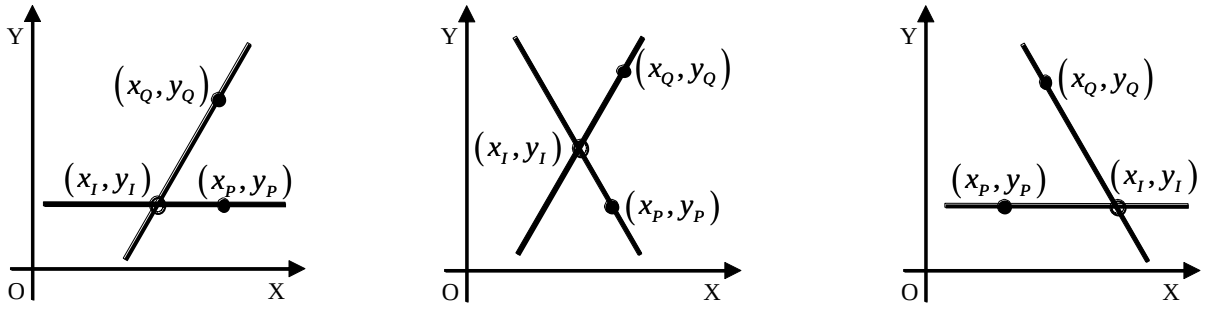
$$\left.\begin{aligned} y_I &= y_P \\ y_I &= \sqrt{3} \cdot (x_I - x_Q) + y_Q \end{aligned}\right\} \Rightarrow (x_I, y_I) = \left(x_Q + \frac{y_P - y_Q}{\sqrt{3}}, y_P \right)$$

The intersection of lines with slopes of 60° and 120° (see Figure 8b):

$$\left.\begin{aligned} y_I &= \sqrt{3} \cdot (x_P - x_I) + y_P \\ y_I &= \sqrt{3} \cdot (x_I - x_Q) + y_Q \end{aligned}\right\} \Rightarrow (x_I, y_I) = \left(\frac{x_P + x_Q + \sqrt{3} \cdot (y_P - y_Q)}{2}, \frac{y_P + y_Q + \sqrt{3} \cdot (x_P - x_Q)}{2} \right)$$

And the intersection of lines with slopes of 0° and 120° (see Figure 8c):

$$\left.\begin{aligned} y_I &= y_P \\ y_I &= \sqrt{3} \cdot (x_Q - x_I) + y_Q \end{aligned}\right\} \Rightarrow (x_I, y_I) = \left(x_Q + \frac{y_Q - y_P}{\sqrt{3}}, y_P \right)$$

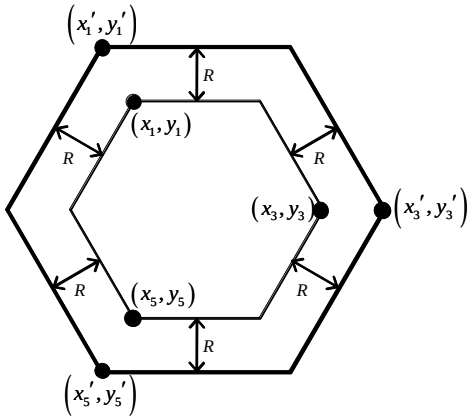


(a) 0° with 60° (b) 60° with 120° (c) 0° with 120°

Figure 8. Intersection of lines with different slopes.

3.2.2. Pseudo-hexagon Extension

In the localization algorithm based on hexagonal intersection we need to extend a pseudo-hexagon by a factor equal to *R*, as we can see in Figure 9a. If the original pseudo-hexagon is defined by vertices (x_1, y_1), (x_3, y_3), and (x_5, y_5), then new vertices of the new pseudo-hexagon extending by a factor equal to *R* are shown in Figure 9b.



(a) Extending a pseudo-hexagon

$$(x_1', y_1') = \left(x_1 - \frac{R}{\sqrt{3}}, y_1 + R \right)$$

$$(x_3', y_3') = \left(x_3 + \frac{2R}{\sqrt{3}}, y_3 \right)$$

$$(x_5', y_5') = \left(x_5 - \frac{R}{\sqrt{3}}, y_5 - R \right)$$

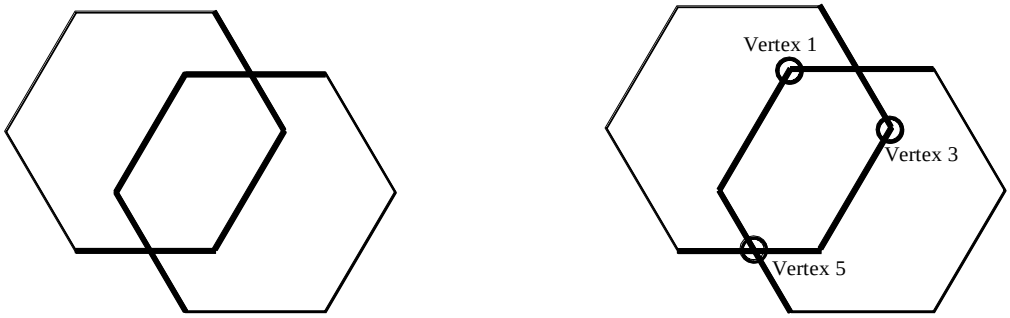
(b) Expressions for the new vertices

Figure 9. Pseudo-hexagon extension.

3.2.3. Pseudo-hexagon Intersection

To intersect two pseudo-hexagons, the following process is carried out:

1. Select the six sides closer to the center of the pseudo-hexagons (Figure 10a).
2. Compute nine intersections of lines.
3. Select vertices 1, 3, and 5 defining the intersection pseudo-hexagon (Figure 10b).



(a) Selecting sides closer to the center (b) Obtaining vertices 1, 3, and 5 of the intersection

Figure 10. Pseudo-hexagons intersection.

Selecting internal lines

Definition: Given a hexagon or_pseudo-hexagon, the side going from vertex *i* to vertex *j* is ij.

Definition: Given two parallel lines P and Q, P is above Q (and Q is below P) if $y_P > y_Q$ at $x = 0$. Figure 11 shows the three possible situations.

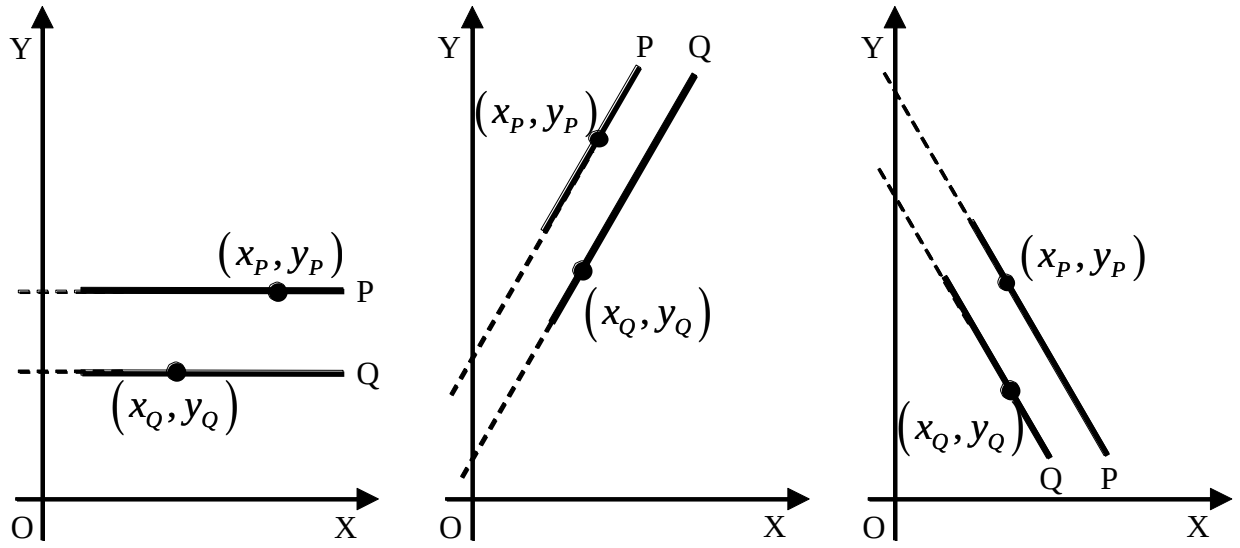


Figure 11. Parallel lines with different slopes.

To get the sides that are closer to the center of two pseudo-hexagons A and B, we will compare each side of A with the corresponding side of B, and select the appropriate in every case:

- Sides $\overline{12}, \overline{23}, \text{and } \overline{61}$: we select those that are below (Figure 12a, 12b, and 12f).
- Sides $\overline{34}, \overline{45}, \text{and } \overline{56}$: we select those that are above (Figure 12b, 12c, and 12d).

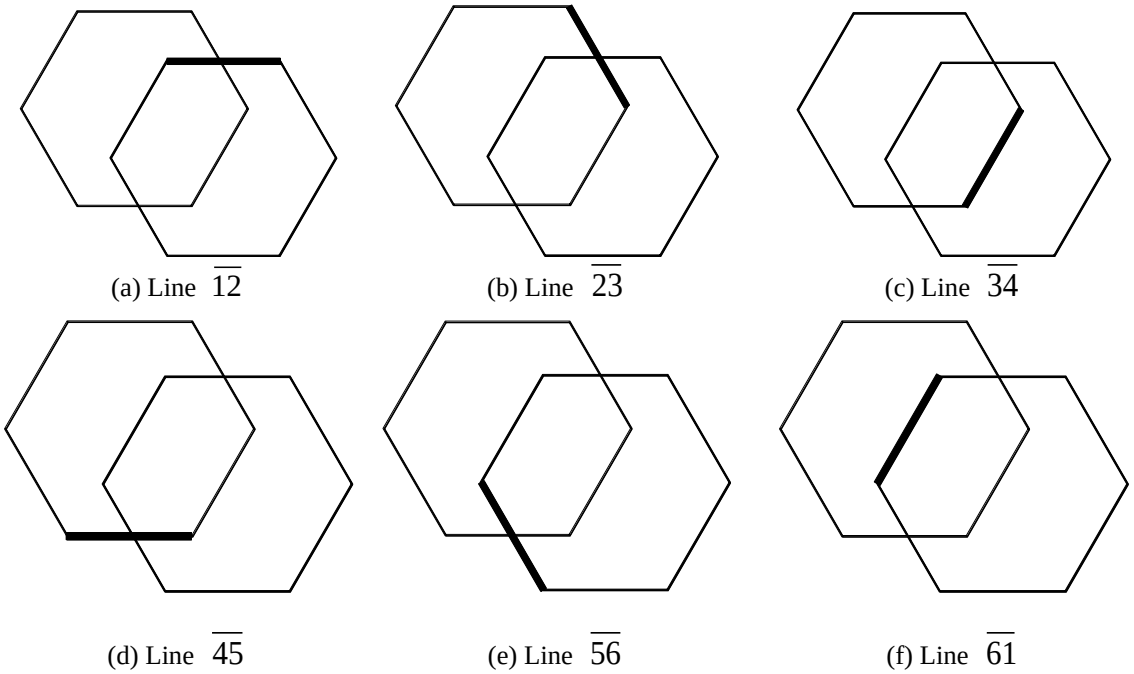


Figure 12. Selecting internal lines.

Calculating intersections and selecting vertices 1, 3, and 5

After selecting the internal lines, we must get nine cut points because there are nine intersections of the selected lines that could be a vertex 1, 3, or 5. There are three candidates to vertex 1 of the intersection, three candidates to vertex 3, and three candidates to vertex 5. Next, we detail the possible candidates and the correct point in each case.

- **Vertex 1**. Figure 13 shows that we have to calculate the cut points 12×56, 12×61, and 23×61, and select the lower point on the right. Consequently, we select the point or points with the lowest coordinate *y* and then the point with the highest coordinate *x*.

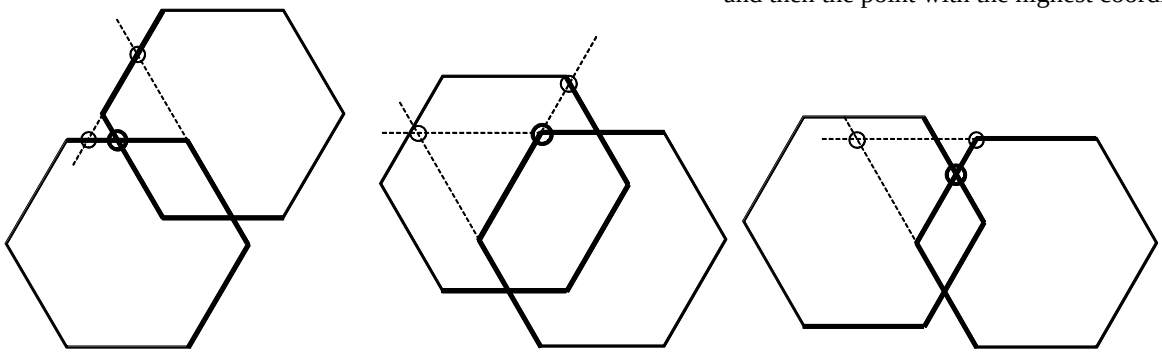


Figure 13. Cases for the vertex 1.

- **Vertex 3**. Figure 14 shows that we have to calculate the cut points 12×34, 23×34, and 23×45, and select the point on the left. Consequently, we select the point with the lowest coordinate *x*.

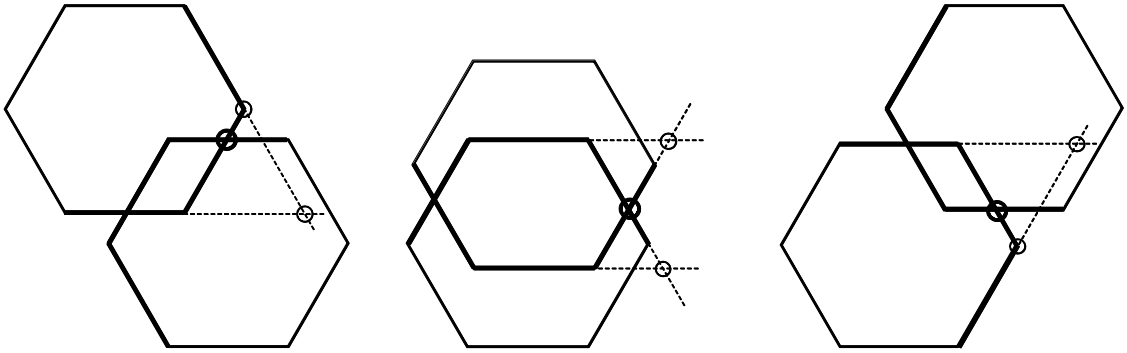


Figure 14. Cases for the vertex 3.

- **Vertex 5**. Figure 15 shows that we have to calculate the cut points 45×56, 45×61, and 34×56, and select the upper point on the right. Consequently, we select the point or points with the highest coordinate *y* and, then, the point with the highest coordinate *x*.

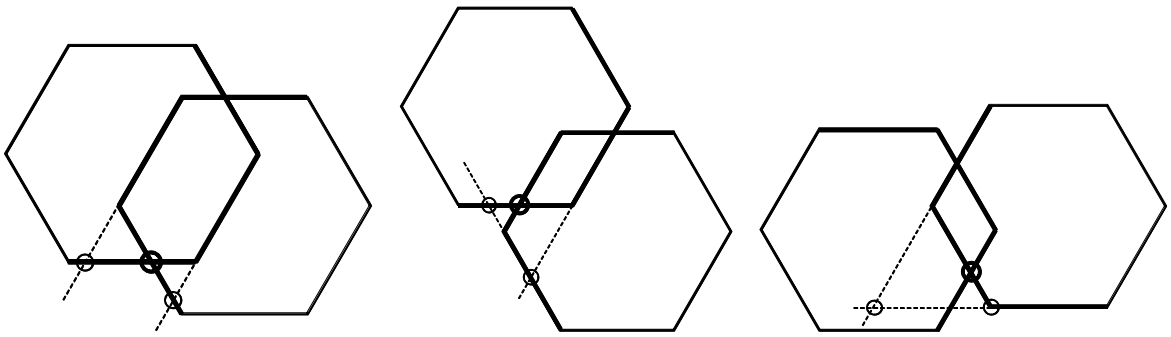


Figure 15. Cases for the vertex 5.

4. SIMULATION ENVIRONMENT

In order to evaluate the new localization proposal, we have used a simulation environment developed for the EIDOS (Equipment Destined for Orientation and Safety) project, which proposes a WSN-based architecture applied to wildfire fighting operations [22]. This environment handles several different types of information (geographical data, vegetation models, fire models, etc.) and devices (sensors, aerial vehicles, and mobile devices). The simulation tool is, therefore, actually composed of several independent and interconnected modules.

Next, we detail the sensor network simulator, which is composed of two parts: the simulation engine and the WSN simulator itself.

4.1. Simulation Engine

The simulation engine has been developed in *Python* [23]. It allows a sensor network simulation to be executed and dynamically controlled in *TOSSIM* (as the next section describes). Moreover, this programming language provides an interface to the *MySQL* database manager [24].

Before running a simulation, several initialization tasks must be performed. In particular, the simulation engine accesses a database to obtain information about the forest area, and the wildfire. Additionally, it deploys a network with the following input parameters:

- Number of nodes
- Percentage of beacon nodes

The position of each node is stored in the database, in order to allow the accuracy of the localization algorithms to be evaluated. Distances between each pair of nodes are also stored in the database in order to make it possible for the network connectivity to be generated starting from a particular signal propagation model, which is explained in the following.

A simple model for the attenuation of signal strength, at distances d much larger than the carrier wavelength, defines the received signal power as proportional to:

$$\frac{1}{d^2} \quad \text{(in free space)}$$

As a concrete example of the above, the Friis free space model [25] defines the receive signal power, P_R for carrier wavelength λ, a receiver with receive antenna gain G_R, at distance d from a transmitter with transmit power P_T and transmit antenna gain G_T, as:

$$P_R = P_T G_T G_R \left(\frac{\lambda}{4\pi R} \right)^n$$

where n is a parameter that can be adapted to obtain certain P_T and P_R. We studied the impact of this parameter bearing in mind that our aim was to implement the localization system on IRIS Crossbow motes, whose P_T and minimum P_R sensitivity were 3 dBm and -101 dBm respectively, and the maximum outdoor range 300 meters. Finally, we concluded that the value of n fulfilling these conditions was 2.4.

In short, the simulation engine generates the network connectivity based on the above model, and using the following input parameters:

- Minimum receive signal power (P_R) in dBm
- Transmit power (P_T) in dBm
- Transmission frequency ($1/\lambda$) in Hz
- Receiver antenna gain (G_R) in dBi
- Transmitter antenna gain (G_T) in dBi

We can therefore force one desired coverage range for all the sensors in the network, and then generate the links between sensors with the appropriate reception power, knowing the distances between each pair of nodes.

Another important issue in wireless sensor simulation is radio frequency noise and interference model. TOSSIM simulates the RF noise and interference a node hears, both as well as outside sources. TOSSIM uses the *Closest Pattern Matching* (CPM) algorithm. CMP takes a noise trace as input and generates a statistical model from it. This model can capture bursts of interference and other correlated phenomena, such that it greatly improves the quality of the RF simulation. Thus, once the network connectivity has been generated, the simulator initializes each node and creates its noise model. Finally, it is randomly activated within the first simulated second.

We assume that a subset of the motes constituting the network will be equipped with a GPS receiver. Before starting the simulation, the simulation engine provides each beacon with its position, modeling in this way the real behavior of a GPS receiver. The simulation engine is able to communicate data to TOSSIM by means of the delivery of radio packets, which are similar to the ones transmitted between motes. These packets can be sent instantaneously, or postponed in time. Therefore, once a mote is activated, its position (if it is a beacon) is obtained from the database. The position packet is transmitted to the mote immediately.

After the network has been activated, the course of time is simulated. The main part of the simulation is performed in TOSSIM, which stores the obtained results in flat text files.

4.2. WSN Simulator

The software developed for the sensor nodes is implemented in *nesC* [26], and executed under the *TinyOS v.2.0.2* operating system [27]. This system incorporates a native simulator called *TOSSIM* (*TinyOS SIMulator*) [28], which is able to simulate the execution of the code incorporated by the real mote without the need of additional changes. In this way, once the code has been debugged, it can be directly downloaded into the sensors memory. Also, TOSSIM allows simulating networks composed of several thousands of sensor nodes. For these reasons, TOSSIM is the most suitable option for our purposes.

The TinyOS application incorporates a component modeling a GPS receiver. If this component receives a radio packet delivered by the simulation engine, then the mote concludes that it is a beacon, and it updates its position with the received information.

One of the main components is the localization module, which is responsible for executing the distributed process by means of which sensor motes are able to localize themselves. This process is started once the network has been deployed. Each time the position of a mote is updated, it will be written in a flat text file. The storage of this information on the database will be carried out by the simulation engine at the end of the simulation, allowing the subsequent analysis of the behavior of the localization algorithms.

5. PERFORMANCE EVALUATION

In this section we present the simulation study that was undertaken to evaluate the localization technique described in this work. For performance assessment our proposal was compared to the improved rectangular intersection algorithm [14], detailed in section 2. In this algorithm, each network node distributes the two corners determining the rectangle obtained by intersecting the rectangular areas received from other neighbor nodes. In particular, for each localization method we measure the average error, its accuracy and also the average size of the localization area. Before presenting the simulation results, the experiment setup is outlined.

5.1. Experiment Setup

For the experiment setup we have carried out extensive numerical configurations. In all cases, network nodes were distributed randomly in a square area of 500×500 meters. Regarding the radio model, the minimum receive signal power was set to -88 dBm and the transmit power to 0 dBm, with the aim of forcing a coverage range of 55 meters. The transmission frequency was set to 2.4 GHz, the receiver antenna gain to 1.2 dBi, and the transmitter antenna gain to 1.2 dBi. These values are characteristic for Crossbow IRIS motes [29].

For the configurations, we varied network size from 400 to 1600 nodes. In the simulation results shown, the horizontal axes correspond to the network degree. This value is calculated as the average number of neighbor nodes for each sensor node. Additionally, the amount of beacons was modified from 1% to 10% of the network nodes.

The metrics used for the performance evaluation are as follows:

1. Average error: given the error in the estimation the Euclidean distance between the real and the estimated positions of a node *S*:

$$Error_S = \sqrt{\left(x_e - x_S\right)^2 + \left(y_e - y_S\right)^2}$$

where $\left(x_S, y_S\right)$ is the real position of the sensor, and $\left(x_e, y_e\right)$ is the estimated position, then the average error is:

$$Average\,error = \frac{\sum_{i=1}^{N} Error_i}{N}$$

where *N* is the network size.

2. Estimation accuracy:

$$Accuracy = \left(1 - \frac{Average\,error}{R}\right) \times 100\%$$

where *R* is the coverage range of the sensor nodes.

3. Average localization area: first, we compute the size of the localization area obtained by each node and, then, we calculate an average considering the network size.

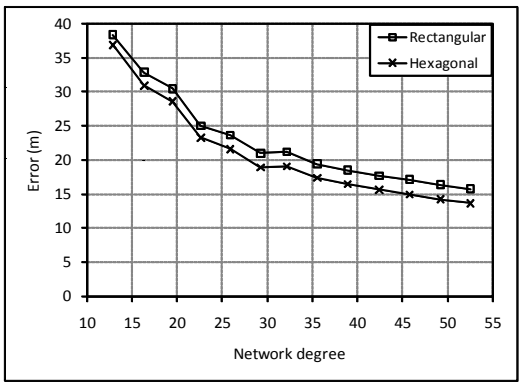
In order to increase the accuracy of the results, each

experiment was repeated 20 times for each configuration, and averages were drawn from the solution set and presented graphically.

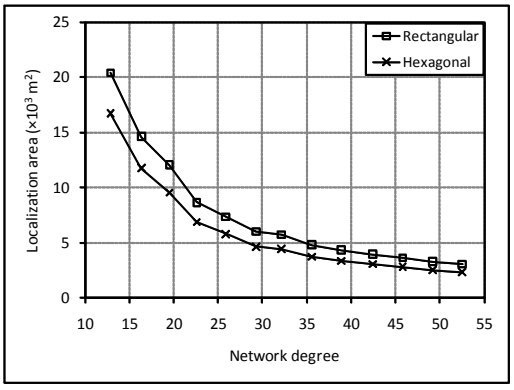
5.2. Experiment Results

Figure 16 shows the average error and the average localization area obtained by both the rectangular intersection and hexagonal intersection algorithms, as a function of the network degree. In this first set of simulations, the percentage of beacons was fixed to 5 percent of the network nodes. Figure 16c and Figure 16e show separately the average error obtained by both algorithms. We can observe that the average error is always lower for the hexagonal intersection proposal. Another conclusion is that, as the network degree increases, the average error decreases for both algorithms, and their corresponding standard deviation.

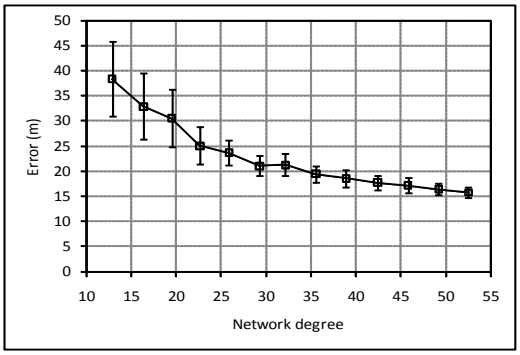
Figure 16d and Figure 16f also present results corresponding to both the rectangular intersection and hexagonal intersection algorithms but, in this case, these are related to the average localization area. The average localization area is always lower for the hexagonal intersection than for the rectangular one.



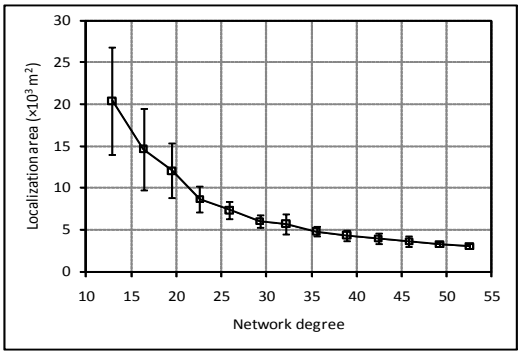
(a) Average error



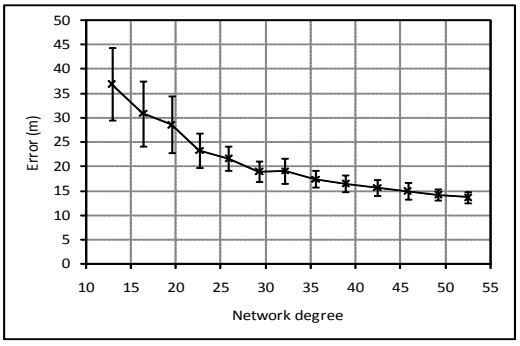
(b) Average localization area



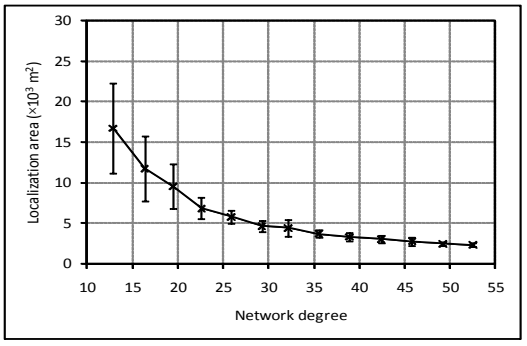
(c) Rectangular error and standard deviation



(d) Rectangular area and standard deviation



(e) Hexagonal error and standard deviation



(f) Hexagonal area and standard deviation

Figure 16. Average error and localization area obtained by the localization algorithms, according to network degree (5% of beacons).

Figure 16a and Figure 16b summarize these results to compare the localization algorithms. In addition, Figure 16a shows that the differences of average error between the algorithms are higher as the network degree increases. In contrast, in Figure 16b, we notice that the differences in terms of localization area are lower when the network degree is higher.

Next, Figure 17 presents the improvement in the estimation error (Figure 17a) and localization area (Figure 17b) obtained when using the localization algorithm based on hexagonal intersection (compared to the rectangular one), when the number of beacons is varied. The conclusions we draw are that the improvement in estimation error is more noticeable as both network degree and number of beacons increase. In contrast, the improvement in the localization area is not affected by the network degree and the amount of beacons in the network.

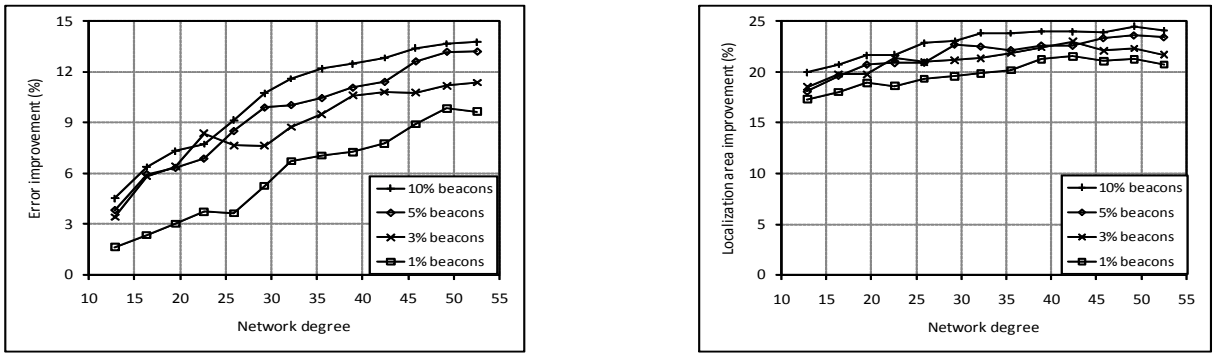
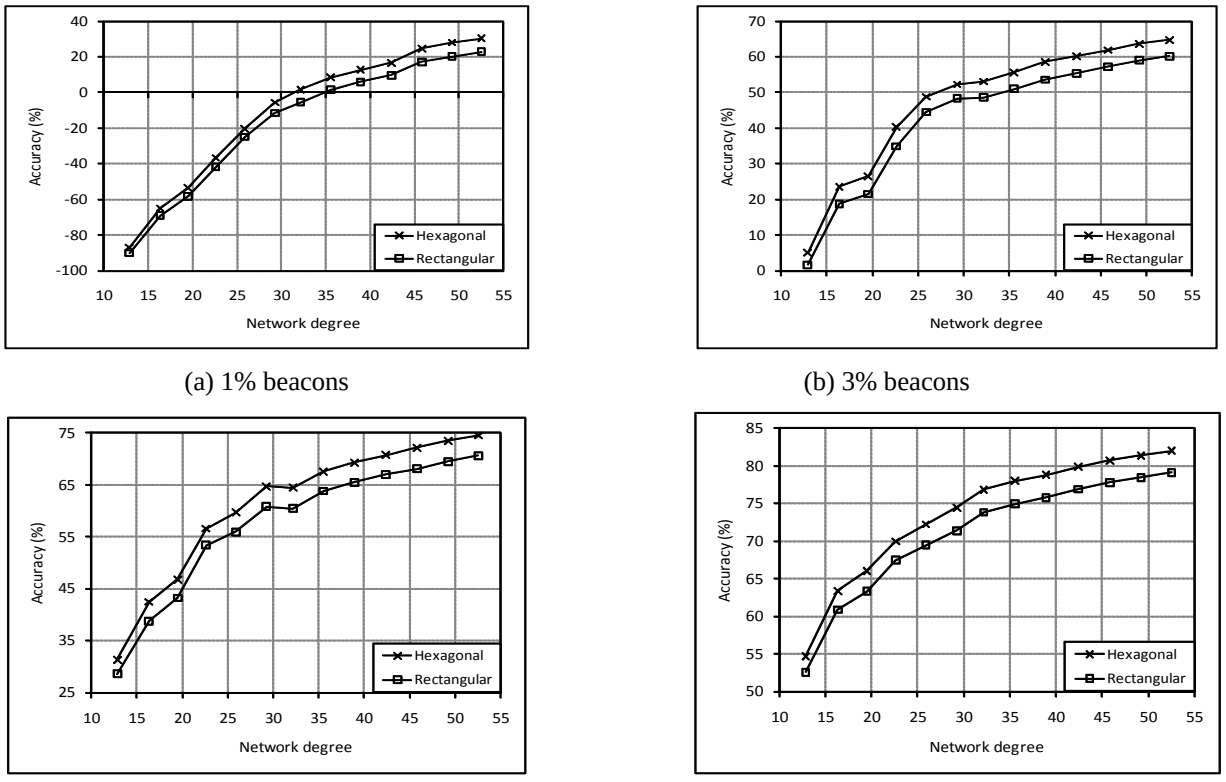


Figure 17. Improvement obtained by the hexagonal intersection algorithm, according to the network degree and the amount of beacon nodes.

To conclude the evaluation, Figure 18 illustrates the accuracy of both the rectangular intersection and hexagonal localization algorithms. We can see that the hexagonal intersection proposal is always more accurate. Obviously, when the percentage of beacon nodes increases, both localization mechanisms are more accurate.



(a) 1% beacons (b) 3% beacons

(c) 5% beacons (d) 10% beacons

Figure 18. Estimation accuracy obtained by the localization algorithms, according to the network degree and the amount of beacon nodes.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a new localization algorithm, called hexagonal intersection, which is suitable for dense wireless sensor networks. This scheme has been subjected to a detailed comparative evaluation, by using a realistic environment. The main advantage of our proposal is that it reduces the error obtained by earlier approaches, which are based on assuming square coverage areas. In addition, the new localization algorithm reduces the size of the estimated localization area, at the same time improving the accuracy.
As future work, we plan to consider the use of mobile beacons to obtain more accurate estimations. We are also interested in evaluating our proposals over a real sensor network prototype.

ACKNOWLEDGMENT

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2009-14475-C04-03" and by the JCCM under grant PII1C09-0101-9476.

REFERENCES

[1] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*, Wiley, 2005.

[2] B. Hoffmann-Wellenhof, H. Lichtenegger, and J. Collins, *GPS: Theory and Practice*. New York: Springer-Verlag, 2001.

[3] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *Proceedings of the 1st ACM international workshop on wireless sensor networks and applications (WSNA '02)*, 2002.

[4] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00)*, 2000.

[5] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)*, 2001.

[6] P. Biswas, H. Aghajan, and Y. Ye, "Integration of angle of arrival information for multimodal sensor network localization using semidefinite programming," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, 2005.

[7] A. Nasipuri and K. Li, "A directionality based location discovery scheme for wireless sensor networks," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[8] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," in *Proceedings of the IEEE INFOCOM*, 2003.

[9] E. Elnahrawy, X. Li, and R. P. Martin, "The limits of localization using signal strength: a comparative study," in *IEEE Sensor and Ad Hoc Communications and Networks*, 2004.

[10] E. M. García, A. Bermúdez, R. Casado, and F. J. Quiles, "Wireless Sensor Network Localization using Hexagonal Intersection," in *Proceedings of the 1st IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07)*, 2007.

[11] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, 7(5), 27–34, 2000.

[12] S. N. Simić and S. Sastry, "A distributed algorithm for localization in random wireless networks" (*unpublished manuscript*), 2002.

[13] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the N-hop multilateration primitive for node localization problems," in *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[14] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem, "Distributed online localization in sensor networks using a moving target," in *Proceedings of the International Symposium of Information Processing Sensor Networks (IPSN'04)*, 2004.

[15] E. M. García, A. Bermúdez, and R. Casado, "Range-Free Localization for Air-Dropped WSNs by Filtering Node Estimation Improvements," in *Proceedings of the IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2010.

[16] E. M. García, A. Bermúdez, and R. Casado, "Range-Free Localization for Air-Dropped WSNs by Filtering Neighborhood Estimation Improvements," in *Proceedings of the 1st International Conference on Computer Science and Information Technology (CCSIT 2011)*, 2011.

[17] J.-P. Sheu, P.-C. Chen, and C.-S. Hsu, "A Distributed Localization Scheme for Wireless Sensor Networks with Improved Grid-Scan and Vector-Based Refinement," *IEEE Transactions on Mobile Computing*, 7(9), 1110–1123, 2008.

[18] J.-P. Sheu, J.-M. Li, and C.-S. Hsu, "A distributed location estimating algorithm for wireless sensor networks," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006.

[19] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proceedings of the 9th ACM Annual International Conference on Mobile Computing and Networking*, 2003.

[20] K.-F. Ssu, C.-H. Ou, and H. C. Jiau, "Localization with mobile anchor points in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, 54(3), 1187–1197, 2005.

[21] B. Xiao, H. Chen, and S. Zhou, "Distributed Localization Using a Moving Beacon in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, 19(5), 587–600, 2008.

[22] E. M. García, M. A. Serna, A. Bermúdez, and R. Casado, "Simulating a WSN-based Wildfire Fighting Support System," in *Proceedings of the IEEE International*

Workshop on Modeling, Analysis and Simulation of Sensor Networks (MASSN'08), in conjunction with the IEEE International Symposium on Parallel and Distributed Processing and Applications (ISPA'08), 2008.

[23] *The Python Programming Language*, http://www.python.org/, 2012.

[24] *MySQL 5.0 Reference Manual*, http://dev.mysql.com/doc/refman/5.0/en/index.html, 2012.

[25] H. T. Friis, "A note on a simple transmission formula," in *Proceedings of the I.R.E. and Waves and Electrons*, 1946.

[26] D. Gay, P. Levis, D. Culler, and E. Brewer, *NesC 1.2 Language Reference Manual*, 2005.

[27] *TinyOS 2.0.2 Documentation*, http://www.tinyos.net/tinyos-2.x/doc/, 2012.

[28] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, 2003.

[29] Crossbow Technology Inc. (now Moog Crossbow), http://www.xbow.com/, 2012.

Authors

Eva María García is an Assistant Professor in Computer Architecture at the Computing Systems Department of the UCLM. He received her M.Sc. degree in Computer Science in 2005 and her M.Sc. degree in Advanced Computer Technologies in 2008. She is also Ph.D. student and her research interests include localization and collaborative information processing algorithms for wireless sensor networks. She is also involved in modeling and routing in interconnection networks.

Rafael Casado received his B.Sc. degree in Computer Science from the University of Castilla-La Mancha in 1993, the M.Sc. degree in Computer Science from the University of Murcia in 1995, and the Ph.D degree in Computer Science from the University of Castilla-La Mancha in 2001. In 1998 he joined the Department of Computer Engineering at the University of Castilla-La Mancha and is an Associate Professor at this department. His research interests include routing and reconfiguration algorithms for high-speed networks and multicomputers, and intelligent collaborative processing in wireless sensor networks. He has participated in more than 30 research projects at national and regional level, conducting several of them. Currently, he is leading a regional project focusing on the use of WSNs to monitor wildfires. He has co-authored more than 30 publications in these areas. Dr. Casado has served as a member of the program committee and reviewer in several conferences and journals, including some of the most prestigious in these areas.

Antonio Robles-Gómez is an Assistant Professor at the Control and Communication Systems Department of the National University of Distance Learning and, also, a researcher at the Computing Systems Department of the University of Castilla-La Mancha. He received hiM.Sc. degree in Computer Science in 2004 and his Ph.D in Computer Science in 2008, both from the University of Castilla-La Mancha. He has several years of experience as a researcher in the interconnect domain. His research interests focus on network modeling and simulation, routing protocols, and network reconfiguration in high-performance networks. He is also involved in the intelligent monitoring of industrial environments by using wireless sensor networks.

Aurelio Bermúdez is an Associate Professor in Computer Architecture at the Computing Systems Department of the UCLM. He received his Ph.D in Computer Science in 2004. His research interests include modeling, routing, and fault-tolerance in interconnection networks, and localization and collaborative processing algorithms for wireless sensor networks. He has co-authored more than 30 publications in these areas.

MªÁngeles Serna received her B.Sc. in Computer Science in 2006 and her M.Sc. degree in Computer Science in 2008, both from the University of Castilla-La Mancha. She is Ph.D. student and her research interests include modeling and routing in interconnection networks. She is also involved in collaborative information processing algorithms for wireless sensor networks.